Vector-Based Prediction of Landing Locations and Early-Zone Rotations in Professional PUBG Mobile Matches on Erangel

Audric Yusuf Maynard Simatupang - 13524010
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13524010@std.stei.itb.ac.id
cubapetuking@gmail.com

***Abstract*—This study proposes a vector- and regression-based approach for predicting where professional PUBG Mobile teams land and how they rotate during the early zones. We use data from PMGC 2025, extracting team positions, safe-zone circle parameters, and flight-path coordinates from match map feeds. For each game, the flight path is modeled as a 2D geometric trajectory on the Erangel map, while team locations are represented as vectors relative to key references such as the flight path and the circle center. These vectorized features are then used to build predictive models for team landing locations and early rotation tendencies. We evaluate the approach by comparing model predictions against the observed movements in the PMGC 2025 Finals, assessing whether the predicted landings and rotations align with actual team behavior. The results suggest that simple geometric representations capture meaningful structure in professional macro-play and can support accurate prediction of both landing points and early rotation patterns.**

***Index Terms*—PUBG Mobile, Esports Analytics, Vector Representation, Geometric Features, Ridge Regression**

I. Introduction

PUBG Mobile is a free-to-play mobile battle royale game available on Android and iOS. In each match, up to 100 players drop onto an island, loot equipment, and compete until only one player or team remains. As one of the most popular games, PUBG Mobile also has an established competitive scene, with its international tournament being the PUBG Mobile Global Championship (PMGC) 2025.

This paper focuses on two early stages: the drop (landing) stage and the initial rotation stage. In casual play, landing and rotation decisions are often made intuitively. In professional tournaments, however, these decisions are strategy-driven. A single poor drop choice or mistimed rotation can lead to an early elimination and, consequently, a significant loss in match points. To reduce this risk, professional teams rely on analysts to study patterns in landing preferences and early movement. This study aims to approximate that analytical process using extracted landing and rotation data, building a vector and regression-based framework to model and predict where teams are likely to land and how they tend to rotate in the early zones.

There are various challenges in building prediction programs for professional PUBG Mobile, with data collection being the main problem. Structured telemetry for team landing locations and early rotations is not publicly available, so the required coordinates must be extracted manually from official tournament map-feed broadcasts. In addition, PUBG Mobile is highly dynamic: variables such as the flight path and safe-zone circle change from game to game, making consistent prediction more difficult. As such, much of the existing discussion is descriptive, while fewer analyses aim to produce quantitative predictions of where teams land or how they rotate.

This paper proposes a vector- and regression-based approach to analyze and predict team landings and early rotations. For landing prediction, each landing point is parameterized relative to the plane trajectory using an along-path component and a signed perpendicular offset. For rotation prediction, team positions are expressed in circle-normalized coordinates relative to the safe-zone center and radius, enabling movement to be modeled in a zone-relative reference frame. Ridge regression is then used to learn team-specific mappings from these geometric representations to predicted landing parameters and next-step rotation displacements.

The study focuses on Erangel matches from PMGC 2025. Flight-path endpoints, team coordinates, and circle parameters are extracted from the tournament map feed and processed into normalized map coordinates. The scope is limited to dropping and early rotation patterns. Combat events, item availability, and vehicle usage are not modeled.

The proposed models will be evaluated by comparing predicted landings and rotation trajectories against observed match data. A game from the PMGC 2025 Finals will be used as a benchmark case, and performance will be assessed using distance-based error metrics in normalized map space.

II. Theoretical Background

This section explains the mathematical foundations used in the study to represent team locations, landing locations, and early rotations geometrically on 2D map. The main idea is to transform raw coordinate data into vectors that are aligned with the plane trajectory and the safe-zone circle. These vectors would then be used by the program to make predictions. These vectors would then be paired with ridge regression to produce quantitative predictions.

A. Vectors in Euclidean Space

All map coordinates are treated as vectors in a 2D Euclidean space. A point on the map is represented as

$$\mathbf{p} = \begin{bmatrix} x \\ y \end{bmatrix},$$

where x and y are normalized coordinates in $[0, 1]$. Using vector notation allows displacements, directions, and distances to be expressed cleanly. For two points **a** and **b**, the displacement from **a** to **b** is $\Delta\mathbf{p} = \mathbf{b} - \mathbf{a}$.

1) 2D Vector Representation: A 2D vector representation provides a consistent way to store and compare team positions across matches. In this work, raw pixel coordinates are converted into normalized coordinates, so every position lies in the same coordinate range regardless of the original screenshot resolution. This makes the analysis independent of the specific pixel dimensions of the source images.

2) Norm, Distance, and Unit Direction: The Euclidean norm of a vector $\mathbf{v} = [v_x, v_y]^T$ is defined as

$$\|\mathbf{v}\| = \sqrt{v_x^2 + v_y^2}.$$

This induces the distance between two points **a** and **b**:

$$d(\mathbf{a}, \mathbf{b}) = \|\mathbf{b} - \mathbf{a}\|.$$

A unit direction vector is obtained by normalization,

$$\hat{\mathbf{v}} = \frac{\mathbf{v}}{\|\mathbf{v}\|},$$

which is used in this paper to define a consistent directional basis aligned with the plane trajectory.

3) Dot Product and Projection: The dot product between **a** and **b** is

$$\mathbf{a} \cdot \mathbf{b} = a_x b_x + a_y b_y.$$

Dot products are useful for measuring alignment and computing projections. The projection of **w** onto a direction **d** can be written as

$$\mathrm{proj}_{\mathbf{d}}(\mathbf{w}) = \frac{\mathbf{w} \cdot \mathbf{d}}{\mathbf{d} \cdot \mathbf{d}}\mathbf{d}.$$

This concept is used implicitly when decomposing a landing point relative to the plane path into along-path and perpendicular components.

B. Geometric Parameterization of the Plane Path

Plane trajectory strongly affects the landing behavior. Therefore, the coordinate system is optimized so the landing is relative to the plane trajectory.

1) Plane Start-End Vector and Direction Basis: Let **s** denote the plane start point and **e** denote the plane end point. The plane direction vector is

$$\mathbf{d} = \mathbf{e} - \mathbf{s},$$

with length

$$L = \|\mathbf{d}\|.$$

When $L > 0$, the unit direction of the plane path is

$$\hat{\mathbf{d}} = \frac{\mathbf{d}}{L}.$$

In the landing model, $\hat{d}_x$, $\hat{d}_y$, and L summarize the geometry of the flight path and are used as regression features.

2) Perpendicular Normal Vector: To measure how far a landing occurs to one side of the flight path, a perpendicular unit normal vector is defined:

$$\hat{\mathbf{n}} = \begin{bmatrix} -\hat{d}_y \\ \hat{d}_x \end{bmatrix}.$$

By construction, $\hat{\mathbf{d}} \cdot \hat{\mathbf{n}} = 0$. Together, $\hat{\mathbf{d}}$ and $\hat{\mathbf{n}}$ form an orthonormal basis aligned with the plane path.

3) Landing Point Parameterization Using (t, u): Let **p** be a team landing point. The landing is parameterized by two scalars: t, the along-path position relative to the plane segment, and u, the signed perpendicular offset from the plane line. Using $\mathbf{d} = \mathbf{e} - \mathbf{s}$, the along-path scalar is

$$t = \frac{(\mathbf{p} - \mathbf{s}) \cdot \mathbf{d}}{\mathbf{d} \cdot \mathbf{d}},$$

and the perpendicular offset is

$$u = (\mathbf{p} - \mathbf{s}) \cdot \hat{\mathbf{n}}.$$

In this work, ridge regression predicts t and u. The predicted landing coordinate can then be reconstructed as

$$\hat{\mathbf{p}} = \mathbf{s} + t\,\mathbf{d} + u\,\hat{\mathbf{n}}.$$

C. Circle Geometry and Circle-Relative Coordinates

Early rotations are strongly influenced by the safe-zone circle. To compare rotations across games with different circle sizes and locations, positions are expressed relative to the circle center and scaled by its radius.

1) Circle Center and Radius: A safe-zone is modeled as a circle with center

$$\mathbf{c} = \begin{bmatrix} c_x \\ c_y \end{bmatrix}$$

and radius r. All quantities are represented in the same normalized coordinate system, so $c_x, c_y \in [0, 1]$, and r is expressed in normalized map units.

2) Normalized Circle-Relative Position $\mathbf{q} = (\mathbf{p}-\mathbf{c})/r$: Given a team position **p**, the circle-relative coordinate is defined as

$$\mathbf{q} = \frac{\mathbf{p} - \mathbf{c}}{r} = \begin{bmatrix} \frac{x - c_x}{r} \\ \frac{y - c_y}{r} \end{bmatrix}.$$

This transformation centers the coordinate system at the circle center and normalizes distance by the circle radius. As a result, positions become more comparable across matches with different circle geometries.

3) Radial Distance $\rho = \|\mathbf{q}\|$: A scalar summary of circle-relative position is the radial distance

$$\rho = \|\mathbf{q}\| = \sqrt{q_x^2 + q_y^2}.$$

When $\rho < 1$, the position lies inside the circle; when $\rho = 1$, it lies on the boundary; and when $\rho > 1$, it lies outside. In this paper, ρ is used as an additional feature to capture how far a team is from the zone in a scale-invariant manner.

D. Systems of Linear Equations and Ridge Regression

After defining geometric feature representations for landing and rotation, prediction is performed using linear models. Ridge regression is selected as a simple baseline that remains stable under limited samples and correlated features.

1) Least Squares as Linear System: Let $X \in \mathbb{R}^{n\times k}$ be a feature matrix and $\mathbf{y} \in \mathbb{R}^n$ be a target vector. A linear model assumes

$$\mathbf{y} \approx X\boldsymbol{\beta},$$

where $\boldsymbol{\beta}$ is the coefficient vector. Ordinary least squares estimates $\boldsymbol{\beta}$ by minimizing

$$\min_{\boldsymbol{\beta}} \|X\boldsymbol{\beta} - \mathbf{y}\|^2.$$

In this work, separate linear models are trained for the landing parameters (t and u) and for rotation displacements (Δq_x and Δq_y).

2) Ridge Regularization λ and Stability: Ridge regression augments least squares with an L_2 penalty:

$$\min_{\boldsymbol{\beta}} \|X\boldsymbol{\beta} - \mathbf{y}\|^2 + \lambda\|\boldsymbol{\beta}\|^2,$$

where $\lambda \geq 0$ controls the strength of regularization. This reduces coefficient magnitude and improves generalization, especially when the dataset is small. The intercept (bias) term can be excluded from penalization, which is commonly done to avoid forcing the baseline prediction toward zero.

3) Closed-Form Solution $(X^TX+\lambda I)^{-1}X^Ty$: Ridge regression has the closed-form solution

$$\boldsymbol{\beta} = (X^TX + \lambda I)^{-1}X^T\mathbf{y}.$$

In implementation, $\boldsymbol{\beta}$ is computed by solving the linear system rather than explicitly taking a matrix inverse. The same formulation is applied to train a global model using all teams and team-specific models when sufficient samples are available.

III. Methodology

This section describes the end-to-end pipeline used in this study, starting from manual data collection on the PMGC 2025 Erangel map feed, followed by preprocessing into learning-ready representations, model training with ridge regression, and finally evaluation using a small-scale Grand Finals case study. The methodology is divided into three parts: (1) data and preprocessing, (2) training and model generation, and (3) evaluation protocol.

A. Data and Preprocessing

The purpose of this stage is to convert raw broadcast observations (pixel coordinates of teams, plane endpoints, and safe-zone circles) into consistent, comparable geometric variables that can be used for learning across different games. All computations assume a fixed 2D coordinate system in which a point on the map is represented as a vector $\mathbf{p} = [x, y]^T$.

1) Data Source: All data is collected manually from the official PMGC 2025 match map feed for the Erangel map. Each record is tagged with `match_id` (match identifier) and `game_no` (game index inside the match). Teams are referenced by `team_name`. For rotation snapshots, time is stored as `t_sec` (seconds since match start following the broadcast timestamp) and `circle_phase` (phase index of the current safe-zone).

All coordinates are first annotated in pixels on the broadcast map panel, then converted into normalized coordinates in $[0,1]\times[0,1]$ using the panel resolution (W, H):

$$x = \frac{x_{\text{px}}}{W}, \qquad y = \frac{y_{\text{px}}}{H}.$$

This normalization ensures that all games share the same numeric scale, independent of the screenshot resolution used during annotation.

2) Landing Data Processing:

a) Main idea: Landing behavior is strongly constrained by the plane trajectory. Therefore, each landing is not treated simply as an absolute point on the map, but as a point relative to the plane segment. The key representation is the (t, u) parameterization: t describes how far along the plane segment a landing occurs, while u describes the signed perpendicular offset from the plane line. This representation is designed to be stable across games with different flight paths.

b) Landing CSV format: The landing dataset stores one row per team per game. Each row contains the plane endpoints and one observed team landing point. In normalized form, the fields are:

- `match_id`, `game_no`, `team_name`
- `plane_start_x`, `plane_start_y`, `plane_end_x`, `plane_end_y`
- `landing_x`, `landing_y`

If the extraction is performed in pixels first, the same structure can be used with pixel values and then normalized before processing.

c) Landing preprocessing step: convert (x, y) to (t, u): Let $\mathbf{s} = [x_s, y_s]^T$ be the plane start, $\mathbf{e} = [x_e, y_e]^T$ the plane end, and $\mathbf{p} = [x_\ell, y_\ell]^T$ the landing point. Define the plane direction $\mathbf{d} = \mathbf{e} - \mathbf{s}$, its length $L = \|\mathbf{d}\|$, and the unit direction $\hat{\mathbf{d}} = \mathbf{d}/L$. The perpendicular unit vector is $\hat{\mathbf{n}} = [-\hat{d}_y, \hat{d}_x]^T$. The landing parameters are computed by:

$$t = \frac{(\mathbf{p}-\mathbf{s})\cdot\mathbf{d}}{\mathbf{d}\cdot\mathbf{d}}, \qquad u = (\mathbf{p}-\mathbf{s})\cdot\hat{\mathbf{n}}.$$

After preprocessing, each landing sample can be represented compactly by (t, u) alongside the original identifiers.

3) Rotation Data Processing:

a) Main idea: Early rotations are strongly influenced by the current safe-zone circle, but circle centers and radii differ across games. To make movement comparable across matches, each team position is converted into a circle-relative coordinate $\mathbf{q} = (\mathbf{p} - \mathbf{c})/r$, where $\mathbf{c}$ is the circle center and r is the circle radius. Rotation learning is then framed as predicting the displacement $\Delta\mathbf{q}$ between consecutive timestamps within the same circle phase.

b) Rotation CSV format: The rotation dataset stores one row per team snapshot at a specific time. Each row contains the team position and the circle parameters at that moment:

- `match_id`, `game_no`, `team_name`
- `t_sec`, `circle_phase`
- `pos_x`, `pos_y`
- `circle_x`, `circle_y`, `circle_r`

As with landing, pixel annotations can be normalized using (W, H) during preprocessing.

c) Convert absolute coordinates to circle-relative coordinates: Given a team position $\mathbf{p} = [x, y]^T$, circle center $\mathbf{c} = [c_x, c_y]^T$, and radius r, the circle-relative coordinate is:

$$\mathbf{q} = \frac{\mathbf{p} - \mathbf{c}}{r} = \begin{bmatrix} \frac{x - c_x}{r} \\ \frac{y - c_y}{r} \end{bmatrix}, \qquad \rho = \|\mathbf{q}\|.$$

Here, ρ summarizes how far the team is from the circle center in a scale-invariant manner ($\rho < 1$ indicates inside the circle).

d) Grouping and pairing by time: Rotation training requires consecutive snapshots. Rows are grouped by (`match_id`, `game_no`, `team_name`, `circle_phase`) and sorted by `t_sec`. Consecutive timestamps (t_i, t_{i+1}) form a valid pair only if $\Delta t = t_{i+1} - t_i > 0$. No cross-phase pairing is performed to ensure both samples share the same safe-zone reference frame.

4) Data Validation and Filtering: To reduce invalid or numerically unstable samples, the following checks are applied during preprocessing:

- **Plane-path degeneracy:** discard landing rows where $\|\mathbf{e} - \mathbf{s}\|$ is too small, since (t, u) becomes unstable.
- **Circle validity:** discard rotation rows where $r \leq 0$.
- **Missing values:** discard rows with empty numeric fields.
- **Pair validity:** discard rotation pairs where $\Delta t \leq 0$.

5) Saved Models into JSON: After training, model parameters are serialized into JSON so inference can be executed without retraining. Two separate JSON artifacts are produced: one for landing and one for rotation.

- **Landing JSON:** stores the selected feature names, ridge hyperparameter λ, sample counts, and coefficient vectors for predicting t and u (global coefficients and optional per-team coefficients when available).
- **Rotation JSON:** stores the selected feature names, ridge hyperparameter λ, pair counts, and coefficient vectors for predicting Δq_x and Δq_y (global coefficients and optional per-team coefficients when available).

B. Training and Model Generation

This stage takes the preprocessed representations and learns regression mappings using ridge regression. The training output is a set of coefficient vectors that can be applied directly during inference.

1) Landing Model Generation:

a) Overview: The landing learning task predicts where a team lands relative to the plane trajectory. Instead of predicting (x, y) directly, the model predicts (t, u), which parameterizes the landing in the plane-aligned coordinate frame. Two separate ridge regressors are trained: one for t and one for u. Predicted (t, u) can be reconstructed back into a landing coordinate $\hat{\mathbf{p}}$ using the plane geometry.

b) Feature construction: For each game, the plane path is defined by start $\mathbf{s}$ and end $\mathbf{e}$. Define $\mathbf{d} = \mathbf{e} - \mathbf{s}$, $L = \|\mathbf{d}\|$, and $\hat{\mathbf{d}} = \mathbf{d}/L$. The landing feature vector is derived only from the plane geometry:

$$\mathbf{x}_{\text{land}} = \begin{bmatrix} 1 & \hat{d}_x & \hat{d}_y & L \end{bmatrix}^T,$$

where the leading 1 is a bias term. This feature choice encodes the orientation and length of the flight path, which are the main match conditions that affect landing feasibility.

c) Model training (global and per-team): Two ridge regression models are fit for landing:

- A model for t: $\hat{t} \approx \mathbf{x}_{\text{land}}^T \boldsymbol{\beta}_t$
- A model for u: $\hat{u} \approx \mathbf{x}_{\text{land}}^T \boldsymbol{\beta}_u$

A **global** landing model is trained using all landing samples pooled across teams. Additionally, when a team has enough landing samples, a **per-team** model is trained for that team using only its samples. During inference, if a team-specific model exists it is used; otherwise, the global model is used as a fallback. This design allows the pipeline to capture both general tendencies and team-specific preferences while remaining stable under limited data.

2) Rotation Model Generation:

a) Overview: The rotation learning task predicts how a team moves during early zones given the current safe-zone circle. Positions are represented in a circle-relative frame $\mathbf{q} = (\mathbf{p} - \mathbf{c})/r$. The model predicts the next-step displacement $\Delta\mathbf{q}$ between two consecutive timestamps within the same phase. Two separate ridge regressors are trained: one for Δq_x and one for Δq_y. Predicted displacements can be accumulated to produce a predicted trajectory in the circle-relative space, then mapped back to absolute map coordinates using $\mathbf{p} = \mathbf{c} + r\mathbf{q}$.

b) Feature construction: For a paired example at time t, the current circle-relative coordinate is $\mathbf{q}_t = [q_x, q_y]^T$, and $\rho = \|\mathbf{q}_t\|$. The elapsed time between the paired samples is $\Delta t_{\min} = \Delta t/60$. The rotation feature vector is:

$$\mathbf{x}_{\text{rot}} = \begin{bmatrix} 1 & q_x & q_y & \rho & \Delta t_{\min} \end{bmatrix}^T.$$

These features represent the team's current relative location, its urgency with respect to the safe-zone (via ρ), and the time interval over which the movement occurs.

c) Model training (global and per-team): For each paired sample, the regression targets are:

$$\Delta q_x = q_{x,2} - q_{x,1}, \qquad \Delta q_y = q_{y,2} - q_{y,1}.$$

Two ridge regression models are trained:

- A model for Δq_x: $\widehat{\Delta q_x} \approx \mathbf{x}_{\text{rot}}^T \boldsymbol{\beta}_{\Delta x}$
- A model for Δq_y: $\widehat{\Delta q_y} \approx \mathbf{x}_{\text{rot}}^T \boldsymbol{\beta}_{\Delta y}$

A **global** rotation model is trained using all valid time-step pairs pooled across teams. When a team has enough consecutive pairs, a **per-team** model is trained using only that team's rotation pairs. During inference, the pipeline selects the team-specific coefficients if available; otherwise, it falls back to the global coefficients. As in landing, this balances specialization (capturing a team's macro tendencies) with robustness (avoiding unstable fits when data is sparse).

C. Evaluation Protocol

To evaluate the proposed models, we perform a small-scale case study using a single game from the PMGC 2025 Grand Finals. From this game, three teams are selected as evaluation subjects. For each selected team, the program is tested using the corresponding match inputs (plane path, circle parameters, and timestamps) while the ground-truth values are obtained by manual observation from the official map feed (recorded as pixel coordinates on the same base map).

The evaluation follows this flow:

1) Select one game from the PMGC 2025 Grand Finals.
2) Select three teams from that game.
3) For each selected team, run the landing and rotation prediction using the program inputs.
4) Record the predicted outputs in pixel coordinates.
5) Record the observed (ground-truth) values in pixel coordinates from the map feed.
6) Compute the prediction error as the pixel-distance difference between prediction and observation.

1) Landing Error Metrics (Pixel Euclidean Distance): Landing predictions are evaluated by comparing the predicted landing point $\hat{\mathbf{p}}_{\text{px}}$ to the manually observed landing point $\mathbf{p}_{\text{px}}$, both expressed in pixels on the same base-map resolution. The landing error is measured using Euclidean distance:

$$\text{Err}_{\text{land,px}} = \|\hat{\mathbf{p}}_{\text{px}} - \mathbf{p}_{\text{px}}\|.$$

For the three selected teams, errors are reported per team, and a simple summary (e.g., mean and maximum error) is also provided for readability.

2) Rotation Error Metrics and Inside-Zone Check (Pixel Space): Rotation predictions are evaluated at the chosen timestamps by comparing the predicted team position $\hat{\mathbf{p}}_{\text{px},t}$ against the manually observed position $\mathbf{p}_{\text{px},t}$, both in pixels:

$$\text{Err}_{\text{rot,px}} = \|\hat{\mathbf{p}}_{\text{px},t} - \mathbf{p}_{\text{px},t}\|.$$

Optionally, a qualitative check is also reported: whether the predicted position is inside the current safe-zone. This can be checked in the circle-relative frame by testing $\|\hat{\mathbf{q}}\| \leq 1$, where $\hat{\mathbf{q}} = (\hat{\mathbf{p}} - \mathbf{c})/r$. This inside-zone check does not replace the pixel error, but helps interpret whether the predicted rotation is at least directionally consistent with the zone objective.

D. Landing Prediction Results

1) Testing Data and Result Summary: For the landing test, the plane trajectory is defined by the endpoints:

$$(500, 1390) \rightarrow (700, 140).$$

For each team, the model outputs one predicted landing point $\hat{\mathbf{p}}_{\text{px}} = (\hat{x}_{\text{px}}, \hat{y}_{\text{px}})$, and the ground-truth landing point $\mathbf{p}_{\text{px}} = (x_{\text{px}}, y_{\text{px}})$ is obtained by manual observation. The landing prediction results (in pixel coordinates) are summarized below:

- **AE:** Predicted $(1256.28, 1095.04)$, Observed $(1210, 1220)$, Error = 133.25 px.
- **KARA:** Predicted $(1130.33, 983.55)$, Observed $(1100, 1030)$, Error = 55.48 px.
- **DK:** Predicted $(870.59, 623.46)$, Observed $(860, 680)$, Error = 57.52 px.

2) Error Calculation (Pixel Euclidean Distance): Landing error is computed using the Euclidean distance in pixel space:

$$\text{Err}_{\text{land,px}} = \left\|\hat{\mathbf{p}}_{\text{px}} - \mathbf{p}_{\text{px}}\right\| = \sqrt{(\hat{x}_{\text{px}} - x_{\text{px}})^2 + (\hat{y}_{\text{px}} - y_{\text{px}})^2}.$$

Using the values above, the mean landing error across the three teams is 82.08 px, with the largest error occurring on AE (133.25 px).

3) Conclusion for Landing: Overall, the landing model produces reasonable landing estimates for KARA and DK (both around $\approx$ 56 px error), while AE shows a larger deviation. This suggests that the plane-aligned (t, u) parameterization can capture a stable relationship between flight-path geometry and landing tendencies, but accuracy may vary depending on team-specific patterns and how well those patterns are represented in the training data.

E. Rotation Prediction Results

1) Testing Data and Result Summary: For the rotation test, the safe-zone center used in the program input is:

$$\mathbf{c}_{\text{px}} = (680, 1160),$$

and the prediction step is evaluated at $\Delta t = 60$ seconds. For each team, we provide the starting position (at the beginning of the step), the predicted position after 60 seconds, and the observed ground-truth position after 60 seconds. The rotation prediction results (in pixel coordinates) are summarized below:

- **AE:** Start $(1200, 1235)$, Predicted $(1080.00, 1359.61)$, Observed $(1150, 1320)$, Error = 80.43 px.
- **KARA:** Start $(1100, 1250)$, Predicted $(1110.00, 1359.66)$, Observed $(1090, 1270)$, Error = 91.86 px.
- **DK:** Start $(810, 760)$, Predicted $(810.00, 1079.01)$, Observed $(810, 970)$, Error = 109.01 px.

2) Error Calculation (Pixel Euclidean Distance): Rotation error at a predicted timestamp is computed in the same way:

$$\text{Err}_{\text{rot,px}} = \left\|\hat{\mathbf{p}}_{\text{px},t} - \mathbf{p}_{\text{px},t}\right\| = \sqrt{(\hat{x}_{\text{px}} - x_{\text{px}})^2 + (\hat{y}_{\text{px}} - y_{\text{px}})^2}.$$

From the results above, the mean rotation error across the three teams is 93.77 px, with the largest error occurring on DK (109.01 px).

3) Conclusion for Rotation: The rotation model is able to produce movement predictions that are directionally plausible within the 60-second step, but the pixel errors remain noticeable (roughly 80–110 px in this case study). This is expected for a baseline linear model because early rotations are influenced by additional latent factors (vehicles, nearby enemies, loot timing, and shot-calling decisions) that are not represented in the current feature set.

IV. Conclusion and Future Work

A. Conclusion

This study introduced a simple vector- and regression-based pipeline to predict professional PUBG Mobile team landings and early-zone rotations on Erangel using manually extracted PMGC 2025 map-feed data. By representing landings in a plane-aligned (t, u) coordinate frame and representing rotations in a circle-relative frame $\mathbf{q} = (\mathbf{p} - \mathbf{c})/r$, the program was able to produce predictions that were reasonably close to observed behavior in a small Grand Finals case study.

From the evaluation results, the landing model achieved moderate pixel-distance errors for two of the tested teams and a larger error for one team, indicating that the approach captures meaningful structure but still depends heavily on team-specific patterns and how well they are represented in training data. Similarly, the rotation model produced directionally plausible 60-second movement predictions, but the errors remained noticeable, reflecting the fact that early rotations are affected by many unobserved factors (such as combat pressure, vehicles, and temporary holds) that are not encoded in the current feature set. Overall, the program worked fairly well as a baseline: it can *somewhat* predict landings and early rotations using only geometric variables, and it provides a reproducible starting point for more advanced esports macro-play modeling.

B. Future Work

There are several clear directions to improve both accuracy and reliability:

a) Collecting more data and expanding coverage: The current dataset is manually collected, which limits the number of games, teams, plane paths, and circle configurations available for training. Future work should expand the dataset across more PMGC matches and include more timestamps per circle phase to reduce sparsity and improve the stability of team-specific models.

b) More extensive testing and stronger evaluation: The current evaluation is a small-scale case study. Future evaluation should include a larger set of held-out games and teams, and use more systematic metrics (mean/median error, worst-case error, and multi-step trajectory drift). Testing across multiple circle phases and longer time horizons would also better reflect real rotation behavior.

c) Improving data collection and preprocessing quality: Because both inputs and ground-truth values are manually extracted, human annotation error can be a major contributor to pixel error. Future work can improve this by standardizing annotation procedures, using more precise coordinate tools, and adding automated consistency checks (e.g., repeated annotation for the same timestamp to estimate uncertainty). A more robust pipeline for converting broadcast frames into base-map coordinates would also reduce noise before training.

d) Improving algorithms and feature representations: The current ridge regression approach is intentionally simple and primarily geometric. Future work can improve prediction quality by adding richer contextual features (e.g., known team landing history priors, map region clustering, road/vehicle availability proxies, or zone distance bands) and by testing alternative models such as non-linear regressors or sequence models for rotations. Another practical improvement is to predict rotations as multi-step trajectories directly (instead of one-step displacement), which may reduce error accumulation during inference.

Appendix

- **Program GitHub:** https://github.com/alivovue/Makalah_Algeo.git
- **Media and Video (Google Drive):** https://drive.google.com/drive/folders/1FguyZPunbGHyEzM4Gb7mkH_LN572F9MX?usp=sharing

Acknowledgment

I would like to express my sincere gratitude for the opportunities and circumstances that have allowed me to complete this paper for IF2123 Aljabar Linier dan Geometri. I am especially thankful to Dr. Ir. Rinaldi Munir, MT., for his clear and thorough teaching of the course material, which greatly helped me throughout the learning process and the preparation of this work.

I would also like to thank my family for their continuous support, encouragement, and patience during the completion of this paper.

Finally, I would like to congratulate Alpha7 Esports for winning PMGC 2025, which served as an additional motivation and inspiration for this study.

References

[1] Munir, R. (2025). *Aljabar Linier dan Geometri 2025–2026*. Retrieved December 24, 2025, from https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2025-2026/algeo25-26.htm

[2] PUBG MOBILE Esports. (2025). *Official PUBG MOBILE Esports YouTube Channel*. Retrieved December 23, 2025, from https://www.youtube.com/pubgmobileesports

[3] PUBG MOBILE Esports. (2025). *PMGC 2025 Grand Finals Day 1 (YouTube video)*. Retrieved December 23, 2025, from https://www.youtube.com/watch?v=ri6IO95SUzE&t=16755s

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 24 Desember 2025

[illegible]

Audric Yusuf Maynard Simatupang 13524010